2022 江苏省大学生程序设计大赛 题解

2022 Jiangsu Collegiate Programming Contest Tutorial

电子科技大学

UESTC

2022 年 5 月 28 日

JSCPC Jiangsu Collegiate Programming Contest

Problem A - PENTA KILL!

题目大意

- 给定一个击杀序列，判断是否有一个人连续击杀五个不同的人.
- 关键词：模拟

Problem A - PENTA KILL!

题目大意

- 给定一个击杀序列，判断是否有一个人连续击杀五个不同的人.
- 关键词：模拟

- std 为 $O(n^2)$ 的做法，枚举每一次五杀的第一个击杀，判断第一个击杀的击杀者的接下来的四次击杀以及第一次击杀是否均为不同的人即可。值得注意的是其他的击杀并不会打断五杀（包括达成五杀的击杀者的死亡）。这点在样例中也有体现。

Problem A - PENTA KILL!

题目大意

- 给定一个击杀序列，判断是否有一个人连续击杀五个不同的人.
- 关键词：模拟

- std 为 $O(n^2)$ 的做法，枚举每一次五杀的第一个击杀，判断第一个击杀的击杀者的接下来的四次击杀以及第一次击杀是否均为不同的人即可。值得注意的是其他的击杀并不会打断五杀（包括达成五杀的击杀者的死亡）。这点在样例中也有体现。
- 梗：这题是在看 MSI 想到的，样例一为 GALA 第一次达成五杀的击杀顺序，而题面中的 unintended disparity in latency between competing teams 则是本次 MSI 重赛理由的英文原文。

Problem B - Prime Ring Plus

题目大意

- 将 $1 \sim n$ 分成若干个环，每个环上相邻两数之和是质数.
- $1 \le n \le 10^4$.
- 关键词：构造，网络流

Problem B - Prime Ring Plus

- 首先任何一个合法方案里没有长度为奇数的环，因为相邻数奇偶性必须不同（唯一一个偶素数是 2）.

Problem B - Prime Ring Plus

- 首先任何一个合法方案里没有长度为奇数的环，因为相邻数奇偶性必须不同（唯一一个偶素数是 2）.
- 分奇偶建二分图并建立源汇，从源拉容量为 2 的边到奇数，若奇数 i 加偶数 j 为素数则从 i 拉容量为 1 的边到 j，从偶数拉容量为 2 的边到汇.

Problem B - Prime Ring Plus

- 首先任何一个合法方案里没有长度为奇数的环，因为相邻数奇偶性必须不同（唯一一个偶素数是 2）.
- 分奇偶建二分图并建立源汇，从源拉容量为 2 的边到奇数，若奇数 i 加偶数 j 为素数则从 i 拉容量为 1 的边到 j，从偶数拉容量为 2 的边到汇.
- 跑网络流，若满流中间的边可以构成一个所有点度数均为 2 且没有重边的无向图，此即答案.

Problem B - Prime Ring Plus

- 首先任何一个合法方案里没有长度为奇数的环，因为相邻数奇偶性必须不同（唯一一个偶素数是 2）.
- 分奇偶建二分图并建立源汇，从源拉容量为 2 的边到奇数，若奇数 i 加偶数 j 为素数则从 i 拉容量为 1 的边到 j，从偶数拉容量为 2 的边到汇.
- 跑网络流，若满流中间的边可以构成一个所有点度数均为 2 且没有重边的无向图，此即答案.
- 正确性来自于满流方案与合法解一一对应.

Problem C - Jump and Treasure

题目大意

- 从坐标为 0 的位置出发从左往右跳，只能跳到整数坐标，跳跃的距离不能超过 p;
- 跳到第 i 个位置获得 a_i 个金币，$[n+1,+\infty)$ 是终点;
- 第 x 关只能跳到 x 的倍数的位置;
- Q 个询问，每次询问在某个关卡中获得的最多金币数是多少.
- 关键词：DP，单调队列，调和级数

Problem C - Jump and Treasure

- 先考虑在第 1 关怎么做，能跳的距离不超过 p，假设走到位置 i 的最优总金币数是 $f[i]$，有

Problem C - Jump and Treasure

- 先考虑在第 1 关怎么做，能跳的距离不超过 p，假设走到位置 i 的最优总金币数是 $f[i]$，有

$$f[i] = \max_{i-j \le p} (f[j]) + a[i]$$

Problem C - Jump and Treasure

- 先考虑在第 1 关怎么做，能跳的距离不超过 p，假设走到位置 i 的最优总金币数是 $f[i]$，有

$$f[i] = \max_{i-j \le p} (f[j]) + a[i]$$

- 这是一个经典的单调队列优化 dp 的模型.

Problem C - Jump and Treasure

- 那么在第 k 关，能跳的位置只有 $0, k, 2k...$ 直到 $n+1$ 以后，一共 $n/k+1$ 个位置需要考虑进去。最多一共 n 关，考虑的位置总数是

Problem C - Jump and Treasure

- 那么在第 k 关，能跳的位置只有 $0, k, 2k\dots$ 直到 $n+1$ 以后，一共 $n/k+1$ 个位置需要考虑进去。最多一共 n 关，考虑的位置总数是

$$\sum_{k=1}^{n} \frac{n}{k} + 1 = n + n\sum_{k=1}^{n} \frac{1}{k} = O(n \ln n)$$

Problem C - Jump and Treasure

- 那么在第 k 关，能跳的位置只有 $0, k, 2k \ldots$ 直到 $n+1$ 以后，一共 $n/k+1$ 个位置需要考虑进去。最多一共 n 关，考虑的位置总数是
$$\sum_{k=1}^{n} \frac{n}{k} + 1 = n + n\sum_{k=1}^{n} \frac{1}{k} = O(n \ln n)$$
- $\sum_{k=1}^{n} \frac{1}{k}$ 是调和级数，是 log 级别。上式 n 在 10^6 的情况下大约是 1.5×10^7，单调队列优化 dp 可过，如果用线段树多一个 log 会被卡.
- 总复杂度 $O(n \ln n)$.

Problem D - Finding Pairs

题目大意

- 给定一个序列和一个数 k，每次询问一个区间，问在区间中找若干对不含重复元素的距离为 k 的数的最大权值和.
- 关键词：分块，莫队，DP

Problem D - Finding Pairs

- 考虑莫队.

Problem D - Finding Pairs

- 考虑莫队.
- 首先发现对于下标模 k 余数不同的位置可以分开做.

Problem D - Finding Pairs

- 考虑莫队.
- 首先发现对于下标模 k 余数不同的位置可以分开做.
- 将所有下标模 k 一样的位置提出来，视为一个子序列.

Problem D - Finding Pairs

- 考虑莫队.
- 首先发现对于下标模 k 余数不同的位置可以分开做.
- 将所有下标模 k 一样的位置提出来，视为一个子序列.
- 考虑对每个子序列维护一个 2×2 的 dp 数组，其中 $dp[i][j]$ 表示对于当前区间左边前 i 个不选，右边前 j 个不选的情况下选若干对数所能选出的最大权值和.

Problem D - Finding Pairs

- 考虑莫队.
- 首先发现对于下标模 k 余数不同的位置可以分开做.
- 将所有下标模 k 一样的位置提出来，视为一个子序列.
- 考虑对每个子序列维护一个 2×2 的 dp 数组，其中 $dp[i][j]$ 表示对于当前区间左边前 i 个不选，右边前 j 个不选的情况下选若干对数所能选出的最大权值和.
- 容易发现，在左边/右边添加上一个新的元素的时候都可以 $O(1)$ 更新这个数组，而删除元素则不能。因此回滚莫队即可.

Problem D - Finding Pairs

- 考虑莫队.
- 首先发现对于下标模 k 余数不同的位置可以分开做.
- 将所有下标模 k 一样的位置提出来，视为一个子序列.
- 考虑对每个子序列维护一个 2×2 的 dp 数组，其中 $dp[i][j]$ 表示对于当前区间左边前 i 个不选，右边前 j 个不选的情况下选若干对数所能选出的最大权值和.
- 容易发现，在左边/右边添加上一个新的元素的时候都可以 $O(1)$ 更新这个数组，而删除元素则不能。因此回滚莫队即可.
- 复杂度 $O(n\sqrt{n})$.

Problem D - Finding Pairs

- 考虑莫队.
- 首先发现对于下标模 k 余数不同的位置可以分开做.
- 将所有下标模 k 一样的位置提出来，视为一个子序列.
- 考虑对每个子序列维护一个 2×2 的 dp 数组，其中 $dp[i][j]$ 表示对于当前区间左边前 i 个不选，右边前 j 个不选的情况下选若干对数所能选出的最大权值和.
- 容易发现，在左边/右边添加上一个新的元素的时候都可以 $O(1)$ 更新这个数组，而删除元素则不能。因此回滚莫队即可.
- 复杂度 $O(n\sqrt{n})$.
- 用类似的转移方式可以预处理 $+$ 分块实现在线查询，（或许）常数会小一些，但是还要做一些额外的优化.

Problem E - Playing Cards

题目大意

- 给定两个序列 $\{a_n\}$ 和 $\{b_n\}$，求一个全排列 $\{c_n\}$，
- 求最小的可能的 $\sum_{i=1}^{n} \left\lceil \frac{\max(b_i - a_{c_i}, 0)}{k} \right\rceil$
- 输出方案.
- 关键词：数据结构，贪心

Problem E - Playing Cards

- 考虑一个贪心.

Problem E - Playing Cards

- 考虑一个贪心.
- 首先每次作弊可以视为将 Bob 手中的一张牌减小 k.

Problem E - Playing Cards

- 考虑一个贪心.
- 首先每次作弊可以视为将 Bob 手中的一张牌减小 k.
- 每次对比 Alice 和 Bob 手中牌的最大点数.

Problem E - Playing Cards

- 考虑一个贪心.
- 首先每次作弊可以视为将 Bob 手中的一张牌减小 k.
- 每次对比 Alice 和 Bob 手中牌的最大点数.
- 如果 Bob 手中的最大点数更大，则一定需要作弊一次并将 Bob 手中最大的牌点数减小 k，否则将 Alice 和 Bob 手中最大的牌配成一对并移除.

Problem E - Playing Cards

- 考虑一个贪心.
- 首先每次作弊可以视为将 Bob 手中的一张牌减小 k.
- 每次对比 Alice 和 Bob 手中牌的最大点数.
- 如果 Bob 手中的最大点数更大，则一定需要作弊一次并将 Bob 手中最大的牌点数减小 k，否则将 Alice 和 Bob 手中最大的牌配成一对并移除.
- 用数据结构加速这一过程即可。复杂度 $O(n \log n)$.

Problem E - Playing Cards

- 考虑一个贪心.
- 首先每次作弊可以视为将 Bob 手中的一张牌减小 k.
- 每次对比 Alice 和 Bob 手中牌的最大点数.
- 如果 Bob 手中的最大点数更大，则一定需要作弊一次并将 Bob 手中最大的牌点数减小 k，否则将 Alice 和 Bob 手中最大的牌配成一对并移除.
- 用数据结构加速这一过程即可。复杂度 $O(n \log n)$.
- p.s. 炉石解场想到的题.

Problem F - Pockets

题目大意

- 有 n 种物品和一个容纳重量为 k 的背包，每种物品有价值 v 和重量 w 两种属性，无限个.
- 你可以购买至多 m 次，每次购买一个任意种物品，如果购买了 i 个，最后重量不能超过 $k+i$.
- 一种方案的价值是物品价值之积，求所有方案价值之和 $\bmod 998244353$.
- $n, m, k \leq 10^5$
- 关键词：生成函数，多项式.

Problem F - Pockets

- 考虑一次购买的生成函数

Problem F - Pockets

- 考虑一次购买的生成函数

$$f(x) = \sum_i c_i x^{w_i}$$

Problem F - Pockets

- 考虑一次购买的生成函数

$$f(x) = \sum_i c_i x^{w_i}$$

- 考虑答案就是

Problem F - Pockets

- 考虑一次购买的生成函数

$$f(x) = \sum_i c_i x^{w_i}$$

- 考虑答案就是

$$\sum_{i=0}^{m} \sum_{j=0}^{i+k} [x^j] f^i(x)$$
$$= \sum_{i=0}^{m} [x^{i+k}] f^i(x) \frac{1}{1-x}$$

Problem F - Pockets

- 令 $g(x) = \frac{x}{f(x)}$

Problem F - Pockets

- 令 $g(x) = \frac{x}{f(x)}$
- 原式 $= [x^{m+k}]\frac{f^m(x)}{1-x}\sum_{i=0}^{m} g^{m-i}(x)$

Problem F - Pockets

- 令 $g(x) = \frac{x}{f(x)}$
- 原式 $= [x^{m+k}]\frac{f^m(x)}{1-x}\sum_{i=0}^{m} g^{m-i}(x) = [x^{m+k}]\frac{f^m(x)\times(g^{m+1}(x)-1)}{(1-x)\times(g(x)-1)}$
- 多项式全家桶即可.

Problem F - Pockets

- 令 $g(x) = \frac{x}{f(x)}$
- 原式 $= [x^{m+k}]\frac{f^m(x)}{1-x}\sum_{i=0}^{m} g^{m-i}(x) = [x^{m+k}]\frac{f^m(x)\times(g^{m+1}(x)-1)}{(1-x)\times(g(x)-1)}$
- 多项式全家桶即可.
- 也可以继续化成

Problem F - Pockets

- 令 $g(x) = \frac{x}{f(x)}$
- 原式 $= [x^{m+k}]\frac{f^m(x)}{1-x}\sum_{i=0}^{m} g^{m-i}(x) = [x^{m+k}]\frac{f^m(x)\times(g^{m+1}(x)-1)}{(1-x)\times(g(x)-1)}$
- 多项式全家桶即可.
- 也可以继续化成

$$= [x^{m+k}]\frac{x^{m+1} - f^{m+1}(x)}{(1-x)\times(x-f(x))}$$

Problem F - Pockets

- 令 $g(x)=\frac{x}{f(x)}$
- 原式 $=[x^{m+k}]\frac{f^m(x)}{1-x}\sum_{i=0}^{m} g^{m-i}(x)=[x^{m+k}]\frac{f^m(x)\times(g^{m+1}(x)-1)}{(1-x)\times(g(x)-1)}$
- 多项式全家桶即可.
- 也可以继续化成

$$=[x^{m+k}]\frac{x^{m+1}-f^{m+1}(x)}{(1-x)\times(x-f(x))}$$

- 只需要一次多项式快速幂.

Problem G - GCD on Bipartite Graph

题目大意

- 将 $1 \sim n+m$ 划分到两个集合，一个集合有 n 个元素，另一个有 m 个，
- 要求：从一个集合任选两个数，另一个集合也任选两个数，这四个数的 $\gcd = 1$
- 判断是否有解 + 构造方案
- $1 \le n, m \le 10^5$
- 关键词：数论，构造

Problem G - GCD on Bipartite Graph

- 一个直观的想法：$\{1, 2, 3, 5, 7, 11, \ldots\}$

Problem G - GCD on Bipartite Graph

- 一个直观的想法：$\{1, 2, 3, 5, 7, 11, \ldots\}$
- 故判断条件为 $\min(n, m) \leq sum[n+m] + 1$，其中 $sum[n]$ 表示 $1 \sim n$ 内的质数个数

Problem G - GCD on Bipartite Graph

- 一个直观的想法：$\{1, 2, 3, 5, 7, 11, \ldots\}$
- 故判断条件为 $\min(n, m) \leq sum[n+m]+1$，其中 $sum[n]$ 表示 $1 \sim n$ 内的质数个数
- 在 $\max(n, m)$ 较小的情况下有一定问题

Problem G - GCD on Bipartite Graph

- 一个直观的想法：$\{1, 2, 3, 5, 7, 11, ...\}$
- 故判断条件为 $\min(n, m) \leq sum[n + m] + 1$，其中 $sum[n]$ 表示 $1 \sim n$ 内的质数个数
- 在 $\max(n, m)$ 较小的情况下有一定问题
- 下面证明在 $\max(n, m) \geq 30$ 时上述条件为充要条件

Problem G - GCD on Bipartite Graph

- 不妨设 $n \leq m$.

Problem G - GCD on Bipartite Graph

- 不妨设 $n \le m$.
- 对于每个质数 p，只能有最多一个集合包含至少 2 个 p 的倍数.

Problem G - GCD on Bipartite Graph

- 不妨设 $n \le m$.
- 对于每个质数 p，只能有最多一个集合包含至少 2 个 p 的倍数.
- 不妨设 $p = 2$ 时，集合 2 至少包含 2 个 p 的倍数，此时考虑集合 1 最多放多少数.

Problem G - GCD on Bipartite Graph

- 不妨设 $n \le m$.
- 对于每个质数 p，只能有最多一个集合包含至少 2 个 p 的倍数.
- 不妨设 $p = 2$ 时，集合 2 至少包含 2 个 p 的倍数，此时考虑集合 1 最多放多少数.

Lemma 1

对于所有 $\le \frac{n}{4}$ 的质数，集合 1 最多放一个，其余放集合 2.

Problem G - GCD on Bipartite Graph

- 不妨设 $n \le m$.
- 对于每个质数 p，只能有最多一个集合包含至少 2 个 p 的倍数.
- 不妨设 $p = 2$ 时，集合 2 至少包含 2 个 p 的倍数，此时考虑集合 1 最多放多少数.

Lemma 1

对于所有 $\le \frac{n}{4}$ 的质数，集合 1 最多放一个，其余放集合 2.

- 当 $n + m \ge 30$ 时 $2, 3, 5$ 的倍数大于等于 4 个，故均只能在集合 1 出现 1 次. 此时最优就是使用 $2, 3, 5$.

Problem G - GCD on Bipartite Graph

- 不妨设 $n \le m$.
- 对于每个质数 p，只能有最多一个集合包含至少 2 个 p 的倍数.
- 不妨设 $p = 2$ 时，集合 2 至少包含 2 个 p 的倍数，此时考虑集合 1 最多放多少数.

Lemma 1

对于所有 $\le \dfrac{n}{4}$ 的质数，集合 1 最多放一个，其余放集合 2.

- 当 $n + m \ge 30$ 时 $2, 3, 5$ 的倍数大于等于 4 个，故均只能在集合 1 出现 1 次. 此时最优就是使用 $2, 3, 5$.
- 可以用上述引理推出：如果集合 1 内存在一个不包含 $2, 3, 5$ 作为因子的合数，不能与集合 1 内的任意其他数的 $\gcd > 1$.

Problem G - GCD on Bipartite Graph

- 不妨设 $n \le m$.
- 对于每个质数 p，只能有最多一个集合包含至少 2 个 p 的倍数.
- 不妨设 $p = 2$ 时，集合 2 至少包含 2 个 p 的倍数，此时考虑集合 1 最多放多少数.

Lemma 1

对于所有 $\le \frac{n}{4}$ 的质数，集合 1 最多放一个，其余放集合 2.

- 当 $n + m \ge 30$ 时 $2, 3, 5$ 的倍数大于等于 4 个，故均只能在集合 1 出现 1 次. 此时最优就是使用 $2, 3, 5$.
- 可以用上述引理推出：如果集合 1 内存在一个不包含 $2, 3, 5$ 作为因子的合数，不能与集合 1 内的任意其他数的 $\gcd > 1$.
- 也就能推出集合 1 内的每个质数的倍数只能出现 1 次.

Problem G - GCD on Bipartite Graph

- $n + m \leq 50$ 可以进行爆搜将结果搜出来，其中一份 std 在多组数据下跑了不到 100ms.

Problem G - GCD on Bipartite Graph

- $n + m \leq 50$ 可以进行爆搜将结果搜出来，其中一份 std 在多组数据下跑了不到 100ms.
- $(n, m) = (8, 8), (n, m) = (12, 17), (n, m) = (11, 14 \sim 17)$ 这几种情况需要特判，或者搜出来.

Problem H - Super Gray Pony

题目大意

- 求长为 n 的 $0/1$ 串 S 求 k 次格雷码后的结果，求格雷码是指将 S 看做二进制数，求其在 n 阶格雷码序列中的下标.
- $n \leq 3 \times 10^6, k \leq 10^9$
- **0.5s，64MB**
- 关键词：位运算，lucas 定理，DP，bitset.

Problem H - Super Gray Pony

- 格雷码性质：$Gray_x = x \bigoplus (x >> 1)$.

Problem H - Super Gray Pony

- 格雷码性质：$Gray_x = x \bigoplus (x >> 1)$.
- 考虑每求一次格雷码的贡献，x 的第 i 位变成 $x_i \bigoplus x_{i+1}$.

Problem H - Super Gray Pony

- 格雷码性质：$Gray_x = x \bigoplus (x >> 1)$.
- 考虑每求一次格雷码的贡献，x 的第 i 位变成 $x_i \bigoplus x_{i+1}$.
- 考虑求 k 次，$x_i, x_{i+1}, ..., x_{i+k}$ 贡献到 x 的第 i 位上的异或次数是 $C_k^0, C_k^1, ..., C_k^k$.

Problem H - Super Gray Pony

- 格雷码性质：$Gray_x = x \bigoplus (x >> 1)$.
- 考虑每求一次格雷码的贡献，x 的第 i 位变成 $x_i \bigoplus x_{i+1}$.
- 考虑求 k 次，$x_i, x_{i+1}, ..., x_{i+k}$ 贡献到 x 的第 i 位上的异或次数是 $C_k^0, C_k^1, ..., C_k^k$.
- 求异或只关心系数 $\mod 2$，由 lucas 定理可以推得 $C_k^m \mod 2 = [k \ \& \ m = m]$.

Problem H - Super Gray Pony

- 格雷码性质：$Gray_x = x \bigoplus (x >> 1)$.
- 考虑每求一次格雷码的贡献，x 的第 i 位变成 $x_i \bigoplus x_{i+1}$.
- 考虑求 k 次，$x_i, x_{i+1}, ..., x_{i+k}$ 贡献到 x 的第 i 位上的异或次数是 $C_k^0, C_k^1, ..., C_k^k$.
- 求异或只关心系数 $\bmod 2$，由 lucas 定理可以推得 $C_k^m \mod 2 = [k \ \& \ m = m]$.
- 因此，对于所有的 S_i，我们需要找到：不低于它的位上与它的距离 m 满足 m 为 k 的子集的位，并做异或和.

Problem H - Super Gray Pony

- 格雷码性质：$Gray_x = x \bigoplus (x >> 1)$.
- 考虑每求一次格雷码的贡献，x 的第 i 位变成 $x_i \bigoplus x_{i+1}$.
- 考虑求 k 次，$x_i, x_{i+1}, ..., x_{i+k}$ 贡献到 x 的第 i 位上的异或次数是 $C_k^0, C_k^1, ..., C_k^k$.
- 求异或只关心系数 $\bmod 2$，由 lucas 定理可以推得 $C_k^m \mod 2 = [k \ \& \ m = m]$.
- 因此，对于所有的 S_i，我们需要找到：不低于它的位上与它的距离 m 满足 m 为 k 的子集的位，并做异或和.
- 至此，使用 FFT 做卷积可以做到 $O(n \log n)$，但是不能通过本题.

Problem H - Super Gray Pony

- 考虑对 k 按位 DP，设 $f[b][i]$ 为：只加入 k 的低 b 位，位置 i 上的答案.

Problem H - Super Gray Pony

- 考虑对 k 按位 DP，设 $f[b][i]$ 为：只加入 k 的低 b 位，位置 i 上的答案.
- 逐一考虑 m 每一位的取值：

Problem H - Super Gray Pony

- 考虑对 k 按位 DP，设 $f[b][i]$ 为：只加入 k 的低 b 位，位置 i 上的答案.
- 逐一考虑 m 每一位的取值：
  1. 若 k 的第 b 位为 0，$f[b][i] = f[b-1][i]$

Problem H - Super Gray Pony

- 考虑对 k 按位 DP，设 $f[b][i]$ 为：只加入 k 的低 b 位，位置 i 上的答案.
- 逐一考虑 m 每一位的取值：
  1. 若 k 的第 b 位为 0，$f[b][i] = f[b-1][i]$
  2. 若 k 的第 b 位为 1，则 m 中第 b 位可以取 $0/1$，对应 i，$f[b][i] = f[b-1][i] \bigoplus f[b-1][i+2^{b-1}]$

Problem H - Super Gray Pony

- 考虑对 k 按位 DP，设 $f[b][i]$ 为：只加入 k 的低 b 位，位置 i 上的答案.
- 逐一考虑 m 每一位的取值：
  1. 若 k 的第 b 位为 0，$f[b][i] = f[b-1][i]$
  2. 若 k 的第 b 位为 1，则 m 中第 b 位可以取 0/1，对应 i，$f[b][i] = f[b-1][i] \bigoplus f[b-1][i+2^{b-1}]$
- 可以发现，这个 DP 可以写成 bitset 的形式，复杂度为 $O(\frac{n\log n}{W})$.

Problem I - Cutting Suffix

题目大意

- 把一个字符串的所有后缀划分到两个集合，每个集合非空.
- 最小化：$\sum_{i \in T_1} \sum_{j \in T_2} \mathsf{LCP}(suf_i, suf_j)$.
- $2 \leq |S| \leq 10^5$.
- 关键词：字符串

Problem I - Cutting Suffix

- 如果这个串有两个字母不相同，那就可以把一种字母开头的后缀放一个集合，其他后缀放另一集合，此时任意两个后缀的 LCP $= 0$.

Problem I - Cutting Suffix

- 如果这个串有两个字母不相同，那就可以把一种字母开头的后缀放一个集合，其他后缀放另一集合，此时任意两个后缀的 LCP $= 0$.
- 如果所有字母都相同，那就把最后一个字母放一个集合，其他后缀放另一个集合，此时任意两个后缀的 LCP $= 1$，答案就是 $|S| - 1$.

Problem J - Balanced Tree

题目大意

- 求 n 个点组成的每个节点都满足左右子树大小相差至多 1 的二叉树个数.
- $0 \leq n < 2^{64}$.
- 关键词：计数

Problem J - Balanced Tree

- 设 $f[x]$ 为 n 个点的 Super Balanced Tree 个数，有

Problem J - Balanced Tree

- 设 $f[x]$ 为 n 个点的 Super Balanced Tree 个数，有

$$f[0] = 1$$

$$f[x] = \begin{cases} 2f[\frac{x}{2}] \cdot f[\frac{x}{2} - 1], & x\text{为偶数} \\ f[\frac{x-1}{2}]^2, & x\text{为奇数} \end{cases}$$

Problem J - Balanced Tree

- 设 $f[x]$ 为 n 个点的 Super Balanced Tree 个数，有

$$f[0] = 1$$

$$f[x] = \begin{cases} 2f[\frac{x}{2}] \cdot f[\frac{x}{2} - 1], & x\text{为偶数} \\ f[\frac{x-1}{2}]^2, & x\text{为奇数} \end{cases}$$

- 容易注意到答案是 2 的次幂.

Problem J - Balanced Tree

- 设 $g[x] = \log_2 f[x]$，有

Problem J - Balanced Tree

- 设 $g[x] = \log_2 f[x]$，有

$$g[0] = 0$$

$$g[x] = \begin{cases} g[\frac{x}{2}] + g[\frac{x}{2} - 1] + 1, & x\text{为偶数} \\ 2g[\frac{x-1}{2}], & x\text{为奇数} \end{cases}$$

Problem J - Balanced Tree

- 设 $g[x] = \log_2 f[x]$，有

$$g[0] = 0$$

$$g[x] = \begin{cases} g[\frac{x}{2}] + g[\frac{x}{2} - 1] + 1, & x\text{为偶数} \\ 2g[\frac{x-1}{2}], & x\text{为奇数} \end{cases}$$

- 可以注意到递归下去状态只有 $O(\log n)$ 个，但是时间和空间限制不允许记忆化搜索.

Problem J - Balanced Tree

- 设 $g[n] = a \cdot g[x] + b \cdot g[x-1] + c$，有

Problem J - Balanced Tree

- 设 $g[n] = a \cdot g[x] + b \cdot g[x-1] + c$，有

$g[n] = a \cdot g[x] + b \cdot g[x-1] + c$

Problem J - Balanced Tree

- 设 $g[n] = a \cdot g[x] + b \cdot g[x-1] + c$，有

$$g[n] = a \cdot g[x] + b \cdot g[x-1] + c$$
$$= \begin{cases} a \cdot (g[\frac{x}{2}] + g[\frac{x}{2} - 1] + 1) + b \cdot (2g[\frac{x-1-1}{2}]) + c, & x\text{为偶数} \\ a \cdot (2g[\frac{x-1}{2}]) + b \cdot (g[\frac{x-1}{2}] + g[\frac{x-1}{2} - 1] + 1) + c, & x\text{为奇数} \end{cases}$$

Problem J - Balanced Tree

- 设 $g[n] = a \cdot g[x] + b \cdot g[x-1] + c$，有

$$g[n] = a \cdot g[x] + b \cdot g[x-1] + c$$

$$= \begin{cases} a \cdot (g[\frac{x}{2}] + g[\frac{x}{2} - 1] + 1) + b \cdot (2g[\frac{x-1-1}{2}]) + c, & x\text{为偶数} \\ a \cdot (2g[\frac{x-1}{2}]) + b \cdot (g[\frac{x-1}{2}] + g[\frac{x-1}{2} - 1] + 1) + c, & x\text{为奇数} \end{cases}$$

$$= \begin{cases} a \cdot g[\frac{x}{2}] + (a+2b) \cdot g[\frac{x}{2} - 1] + c + a, & x\text{为偶数} \\ (2a+b) \cdot g[\frac{x-1}{2}] + b \cdot g[\frac{x-1}{2} - 1] + c + b, & x\text{为奇数} \end{cases}$$

Problem J - Balanced Tree

- 初始 $g[n] = 1 \cdot g[n] + 0 \cdot g[n-1] + 0.$

Problem J - Balanced Tree

- 初始 $g[n] = 1 \cdot g[n] + 0 \cdot g[n-1] + 0.$
- 每次 x 除以 2，按上式分奇偶更新 a, b, c，

Problem J - Balanced Tree

- 初始 $g[n] = 1 \cdot g[n] + 0 \cdot g[n-1] + 0$.
- 每次 x 除以 2，按上式分奇偶更新 a, b, c，
- 最终 $x = 1$ 时有 $g[n] = a \cdot g[1] + b \cdot g[0] + c = c$，

Problem J - Balanced Tree

- 初始 $g[n] = 1 \cdot g[n] + 0 \cdot g[n-1] + 0$.
- 每次 x 除以 2，按上式分奇偶更新 a, b, c，
- 最终 $x = 1$ 时有 $g[n] = a \cdot g[1] + b \cdot g[0] + c = c$，
- 答案为 $2^c \mod 2^{64}$.

Problem J - Balanced Tree

- 初始 $g[n] = 1 \cdot g[n] + 0 \cdot g[n-1] + 0$.
- 每次 x 除以 2，按上式分奇偶更新 a, b, c，
- 最终 $x = 1$ 时有 $g[n] = a \cdot g[1] + b \cdot g[0] + c = c$，
- 答案为 $2^c \mod 2^{64}$.
- 所以当 $c \geq 64$ 直接输出 0，否则输出 2^c 即可.

Problem J - Balanced Tree

- 初始 $g[n] = 1 \cdot g[n] + 0 \cdot g[n-1] + 0$.
- 每次 x 除以 2，按上式分奇偶更新 a, b, c，
- 最终 $x = 1$ 时有 $g[n] = a \cdot g[1] + b \cdot g[0] + c = c$，
- 答案为 $2^c \mod 2^{64}$.
- 所以当 $c \geq 64$ 直接输出 0，否则输出 2^c 即可.
- 时间复杂度 $O(T \log n)$，空间复杂度 $O(1)$.

Problem K - aaaaaaaaaaA heH heH nuN

题目大意

- 形如 nunhehhehaaaaa 这种称为优雅串.
- 给定 n，构造恰好有 n 个子序列是优雅串的字符串.
- $0 \leq n \leq 10^9$
- 关键词：构造

Problem K - aaaaaaaaaaA heH heH nuN

- 可以注意到形如 nunhehheh+a$\times x$ 的串，其有效子序列数等于 $2^x - 1$.

Problem K - aaaaaaaaaaA heH heH nuN

- 可以注意到形如 nunhehheh+a$\times x$ 的串，其有效子序列数等于 2^x-1.
- 所以先设一个前缀为 nunhehhe，然后从高往低遍历 x，每当当前的 n 大于等于于 2^x-1 时，则加入若干个 h 直到 n 小于 2^x-1 ，然后加入一个新的 a.

Problem L - Collecting Diamonds

题目大意

- 长度为 n 的只含有 ABC 的字符串.
- 选择连续的三个位置 ABC，如果 A 在奇数位置就删去 AC；否则删去 B.
- 最大化操作数.
- 关键词：思维

Problem L - Collecting Diamonds

- 注意到如果操作删除 B 则当前 ABC 组不可能继续操作，并且也只有删除 B 才能更改后面的 ABC 组的奇偶性.

Problem L - Collecting Diamonds

- 注意到如果操作删除 B 则当前 ABC 组不可能继续操作，并且也只有删除 B 才能更改后面的 ABC 组的奇偶性.
- 所以应该尽量让每个组都操作一次删除 B，并尽量保证在操作删除 B 之前删除尽量多的 AC.

Problem L - Collecting Diamonds

- 注意到如果操作删除 B 则当前 ABC 组不可能继续操作，并且也只有删除 B 才能更改后面的 ABC 组的奇偶性.
- 所以应该尽量让每个组都操作一次删除 B，并尽量保证在操作删除 B 之前删除尽量多的 AC.
- 所以使用一个变量存储下前面一共删除了几次 B，便可以用以计算可以删除多少个 AC.